УДК 004.2

В.И.Жабин
Национальный технический университет Украины «Киевский политехнический институт»
Украина, 03056, г. Киев, пр. Победы, 37

ПОВЫШЕНИЕ ЭФФЕКТИВНОСТИ ПАРАЛЛЕЛЬНОЙ ОБРАБОТКИ ДАННЫХ И ОБЕСПЕЧЕНИЕ ОТКАЗОУСТОЙЧИВОСТИ МУЛЬТИПРОЦЕССОРНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ РЕАЛЬНОГО ВРЕМЕНИ

V.I.Zhabin
National Technical University of Ukraine "Kyiv Polytechnic Institute"
Ukraine, 03056, Kyiv, Av. Peremohy, 37

INCREASE OF PARALLEL DATA PROCESSING EFFECTIVENESS AND CONTROL OF REAL-TIME MULTIPROCESSOR COMPUTER SYSTEM FAILURE SAFETY

В.І.Жабін
Національний технічний університет України «Київський політехнічний інститут»
Україна, 03056, м. Київ, пр. Перемоги, 37

ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ПАРАЛЕЛЬНОЇ ОБРОБКИ ДАНИХ І ЗАБЕЗПЕЧЕННЯ ВІДМОВОСТІЙКОСТІ МУЛЬТИПРОЦЕСОРНИХ ОБЧИСЛЮВАЛЬНИХ СИСТЕМ РЕАЛЬНОГО ЧАСУ

Рассматриваются вопросы обеспечения отказоустойчивости мультипроцессорных систем с динамическим распределением заданий между процессорами под управлением дескрипторов данных. Показана возможность автоматического исправления ошибок, возникающих вследствие отказа процессоров. Предложен программно-аппаратный метод реконфигурации систем, позволяющий сократить расход команд при восстановлении систем.

Ключевые слова: отказоустойчивые системы, реконфигурируемые системы, динамическое распределение заданий, вычисления, управляемые потоком дескрипторов.

Issues of control of failure safety of data-flow multiprocessor systems with dynamic task distribution between processors are considered. Possibility of the automatic recovery of errors appeared due to processor failure is shown. There is proposed a hardware and software system reconfiguration method, which enables to reduce the list of instructions necessary for system recovery.

Keywords: fail-safe system, reconfigurable computer systems, dynamic task distribution, data-flow computation.

Розглядаються питання забезпечення відмовостійкості мультипроцесорних систем з динамічним розпо-ділом завдань між процесорами під керівництвом дескрипторів даних. Показано можливість автома-тичного виправлення помилок, що виникають внаслідок відмов процесорів. Запропоновано програмно-апаратний метод реконфігурації систем, що дозволяє скоротити витрату команд при відновленні систем.

Ключові слова: відмовостійкі системи, реконфігуровані системи, динамічний розподіл завдань, обчислення, що керуються потоком дискрипторів.

Введение

Возрастание сложности задач управления, решаемых в реальном времени, предъявляет все более высокие требования к производительности и надежности вычислительных систем.

Продолжительность решения задач во многом определяется возможностью распараллеливания процессов с целью максимальной загрузки вычислительных средств. Для разработки параллельных программ, кроме языков параллельного программирования (Ada, Java и др.), находят применение различные расширения последовательных языков (например, mpC, C-DVM, HPF, Fortran-DVM), а также библиотеки параллельного программирования (OpenMP, MPI, PVM и др).

Большинство известных технологий программирования относятся к средствам статического распараллеливания вычислений. Основные задачи распараллеливания в этом случае решаются программистом на этапе разработки программ. Статическое распараллеливание может выполняться автоматически с помощью специальных компиляторов, хотя их возможности весьма ограничены.

Один из недостатков средств статического распараллеливания состоит в том, что при анализе алгоритмов не всегда удается выявить параллельные ветви. Это обусловлено целым рядом причин, к основным из которых можно отнести недостаток информации о динамике процессов.

Перспективным подходом, позволяющим устранить ряд недостатков статического планирования и упростить подготовку параллельных вычислений, является разработка средства динамического распараллеливания вычислений. Распределение работ между вычислительными узлами в этом случае осуществляется системой автоматически в процессе решения задач. Реализация данного подхода позволяет выявить параллельные ветви, которые возникают непосредственно в процессе вычислений.

С целью уменьшения затрат времени на динамическое распределение операций при реализации мелкозернистых алгоритмов были предложены модели вычислений под управлением потоков данных (например, [1,2]). Однако с увеличением зернистости алгоритмов существенно возрастают объемы пересылаемых данных, что снижает эффективность такой модели вычислений. *Уменьшение непроизводительных затрат времени в указанном случае является важной задачей повышения эффективности параллельных систем.*

Для систем управления в реальном времени надежность систем и достоверность результатов вычислений является одной из важнейших характеристик. Достоверность получаемой информации определяется соответствием полученных результатов (возможно с ошибками) истинным, то есть полученным без ошибок. Под надежностью системы понимается ее свойство выполнять свои функции, сохраняя во времени значения установленных эксплуатационных показателей в заданных пределах. Количественная оценка надежности связана с понятием отказа. Отказом в системе считается уход хотя бы одного параметра за пределы, заданные техническими условиями. Такими параметрами могут быть, например, производительность, объем памяти и т. д. Соответственно наработка на отказ трактуется как наработка на спад параметра ниже заданного уровня.

Один из эффективных подходов к решению проблемы повышения надежности систем состоит в обеспечении отказоустойчивости путем введения различного рода избыточности (аппаратурной, информационной и временной). Последствия отказов при наличии резервного оборудования могут быть устранены заменой отказавшего

узла. Соответственно повышается наработка на отказ. В процессе функционирования таких систем необходимо обеспечить контроль и диагностику отказов, реконфигурацию и восстановление функций системы.

Программные способы восстановления систем при локализации отказов используют сложные процедуры голосования процессоров с многократным обращением к памяти [3]. Это требует больших временных затрат и может оказаться неприемлемым для систем реального времени, на время обработки информации в которых накладываются ограничения со стороны внешних факторов. *В связи с этим важной задачей является уменьшение времени восстановления систем.*

Обобщая результаты исследований [4-7], в данной работе показана возможность повышения эффективности потоковых вычислений в модульных мультипроцессорных системах при реализации крупнозернистого параллелизма, а также ускорения восстановления систем при отказе процессоров.

Характеристика модульной мультипроцессорной системы

Задачи ускорения вычислений и повышения надежности систем могут быть совместно решены применением мультипроцессорных систем на однотипных процессорных модулях (ПМ). В этом случае имеется потенциальная возможность ускорения преобразования информации путем увеличения количества ПМ, работающих параллельно. В свою очередь, наличие однотипных ПМ позволяет использовать наиболее экономичное скользящее резервирование аппаратуры, необходимое для обеспечения отказоустойчивости систем.

Для конкретности будем рассматривать мультипроцессорную систему с общей шиной (общей системной магистралью). В процессе функционирования системы один из ПМ выполняет функцию управляющего (УПМ), а остальные являются подчиненными (ППМ). Каждый ПМ содержит собственно процессор и локальную память, которые связаны через локальную шину. В состав интерфейса, связывающего ПМ с системной шиной, входят распределенные аппаратные средства реконфигурации, позволяющие уменьшить расход команд при восстановлении системы в случае отказа процессорных модулей. Арбитраж при обращении к общему адресному пространству осуществляется автоматически на аппаратном уровне. Благодаря локальной памяти процессоры могут выполнять свои программы независимо друг от друга, то есть осуществлять параллельную обработку информации. Кроме обращения к общей памяти, каждый ПМ имеет доступ к части локальной памяти других ПМ. УПМ может воздействовать на ППМ (подключать, отключать от общей системной шины, запускать выполнение программ и т.д.) с помощью команд, записываемых в специальный командный регистр (КР), адрес которого принадлежит к части локальной памяти с доступом со стороны системной шины.

Концепция параллельных вычислений, управляемых потоком дескрипторов данных и заданий.

В основу метода положены следующие теоретические положения.

Вычислительный процесс представляется с помощью графа, каждой вершине которого соответствует задание (функция преобразования данных, определенный объем работы), а каждой дуге – поток данных, необходимых для выполнения задания. Исходные данные для каждой задачи подготавливаются на основании графа и представляют собой набор дескрипторов заданий и данных. Дескрипторы формируются без учета числа процессоров в системе.

Функции, соответствующие вершинам графа, являются независимыми и взаимо-действуют только через данные. Для решения задач разрабатывается библиотека реализации функций. В качестве компонентов библиотеки могут использоваться программные модули, функционирующие в заданной операционной среде и позволя-ющие осуществить необходимое для решения задачи преобразование данных.

Задание для i-й вершины графа описывается дескриптором задания

$$\boldsymbol{W}_i = \{N_i, I_i, \boldsymbol{P}_i, Q_i\},$$

где N_i – уникальное имя данного дескриптора; I_i – идентификатор задания (определяет функцию преобразования данных); $\boldsymbol{P}_i=\{p_{ij}\}$ – множество имен выходных данных (соответствуют дугам, выходящим из i-й вершины и входящим в j-ю вершину); Q_i – суммарное число входных потоков данных для i–го задания (число дуг графа, входящих в i-ю вершину).

Имя потока выходных данных p_{ij} может быть представлено кортежем

$$p_{ij} = \langle N_j, n_{ij}, Q_j \rangle,$$

где n_{ij} – имя входа j-й вершины графа, соответствующего дуге D_{ij}.

Поток данных, соответствующий дуге графа, соединяющей j-ю вершину с i-й вершиной, характеризуется дескриптором данных

$$\boldsymbol{D}_{ji} = \{N_i, n_{ji}, Q_i, A_{ji}\}$$

где A_{ji} – элемент адресации данных, определяющий расположение данных в памяти системы.

Из элементов дескрипторов в соответствии с определенной процедурой F формируются заявки на выполнение i-го задания

$$F(\boldsymbol{W}_i, \boldsymbol{D}_i) \rightarrow \boldsymbol{Z}_i = \{I_i, \boldsymbol{P}_i, \boldsymbol{A}_i\},$$

где $\boldsymbol{A}_i = \{\boldsymbol{A}_{ji} | j \in \boldsymbol{J}\}$ – множество элементов адресации данных для i-го задания; $\boldsymbol{D}_i = \{\boldsymbol{D}_{ji} | j \in \boldsymbol{J}\}$ – множество дескрипторов данных для задания $\boldsymbol{W}_i$; $\boldsymbol{J}$ – множество имен вершин графа, связанных выходящими дугами с i-й вершиной.

Для формирования заявок $\boldsymbol{Z}_i$ используется УПМ. Сложность формирования заявок состоит в том, что данные для i–го задания формируются в разные моменты времени и, кроме того, возможно разными ППМ.

Учитывая уникальность имен N_i дескрипторов для каждого задания, условие активизации (готовности) задания можно также представить в виде

$$\omega_i = \begin{cases} 1, \text{ если } C_i = Q_i; \\ 0, \text{ если } C_i \neq Q_i, \end{cases}$$

где C_i – число принятых для формирования заявки дескрипторов с именем N_i.

УПМ создает таблицу (массив объектов) в общей памяти, i-я строка S_i которой имеет вид

$$N_i : S_i = \langle I_i, \boldsymbol{P}_i, \boldsymbol{A}_i, \boldsymbol{L}_i, C_i, Q_i \rangle,$$

где N_i – имя строки; C_i –счетчик дескрипторов, $\boldsymbol{L}_i=\{\delta_{ji}\}$ – признаки наличия поступивших дескрипторов данных. Число признаков в строке S_i равно числу входных потоков данных Q_i.

При поступлении дескрипторов заданий в соответствующие позиции строк таблицы вводятся значения I_i и $\boldsymbol{P}_i$. При поступлении дескрипторов данных в соответствующие позиции строк записываются элементы множества $\boldsymbol{A}_i$ и соответствующие им признаки δ_{ji}. Позиция элементов адресации определяется значениями n_{ji}, присутствующими в дескрипторах заданий.

Полное накопление множества $\boldsymbol{A}_\text{i}$ элементов адресации данных для задания определяется с помощью счетчика C_i. При поступлении любого дескриптора сравниваются значения Q_i и C_i, затем C_i увеличивается на единицу. Равенство указанных значений является условием ω_i активизации заявки.

Свободные ППМ сами обращаются за заявками. После выполнения задания УПМ возвращает полученный дескриптор данных, который принимает участие при формировании новой заявки. При этом обнуляется счетчик C_i, то есть система готова для повторного формирования заявки с данным именем, если это необходимо.

В системе одновременно могут решаться несколько задач. Для этого достаточно, чтобы множества имен дескрипторов для разных задач не пересекались. На стадии формирования заявок между модулями пересылаются только короткие сообщения (дескрипторы). Потоки непосредственно данных перемещаются только на этапе выполнения заданий. В этом случае вычислительным процессом управляют стандартные средства операционной системы.

Исправление ошибок, вызванных отказом процессоров

Поскольку подготовка задач в соответствии с рассматриваемым методом осуществляется без учета числа ПМ, то появляется потенциальная возможность продолжения вычислений при отказе процессоров до тех пор, пока в системе останется хотя бы один исправный ПМ. В последнем случае ПМ, конечно, должен выполнять функции управляющего и подчиненного. Кроме этого, система может иметь на общей шине резервные модули (РПМ), которыми могут заменяться отказавшие модули.

Рассмотрим реализацию указанной возможности при однократном отказе ППМ (по крайней мере, на продолжении цикла восстановления системы).

Общую память можно защитить аппаратурными методами повышения надежнос-ти, которые достаточно хорошо разработаны. Устранение последствий отказов ППМ может быть возложено на УПМ. В случае отказа ППМ во время выполнения задания УПМ должен обеспечить инициализацию повторного выполнения задания в работоспособном ПМ. Следует учитывать, что до отказа ППМ часть дескрипторов данных могла быть передана в УПМ. В этом случае повторное выполнение задания продублирует уже полученные дескрипторы, что приведет к неправильной модификации счетчиков C_i в некоторых строках таблицы формирования заявок, то есть к ошибке вычислений.

Для обеспечения отказоустойчивости ПМ должны обеспечивать ряд специальных функций.

Специальные функции в режиме ППМ. В процессе выполнения задания ППМ возвращает УПМ в общем случае несколько дескрипторов данных $\boldsymbol{D}_{ji}$ согласно числу элементов в множестве Pi. После успешного завершения задания ППМ должен переслать УПМ признак E_i окончания задания.

Специальные функции в режиме УПМ. Данный модуль должен выявлять факт неисправности ППМ и формировать маску для повторного выполнения задания.

Заявка снабжается маской $\boldsymbol{M}_i$, которая формируется процедурой

$$G(\boldsymbol{P}_i, \delta_{ri} \mid r \in \boldsymbol{R}) \to \boldsymbol{M}_i = \{ \overline{\delta_{ri}} \},$$

где $\boldsymbol{R}$ – множество имен вершин графа, связанных входящими дугами с i–й вершиной.

Разряды маски определяют, какие выходные потоки данных необходимо формировать при повторном выполнении заданий.

Механизм исправления ошибки заключается в следующем. После формирования очередной заявки $\boldsymbol{Z}_i$ ее имя N_i заносится в список имен готовых заявок (СГЗ). Заявки выполняются ППМ только из числа указанных в списке СГЗ.

Имя выбранной для выполнения заявки передается из СГЗ в список имен выполняемых заявок (СВЗ). При поступлении признака E_i заявка $\boldsymbol{Z}_i$ считается выполненной, при этом ее имя удаляется из СВЗ.

При решении задачи пустой СГЗ может указывать на различные ситуации в системе. Первая ситуация заключается в том, что выполнение заявок в ППМ не закончено и дескрипторы данных для формирования очередных заявок еще не поступили, но будут получены через определенный промежуток времени. Вторая ситуация связана с отказом ППМ. В этом случае вычислительный процесс не может продолжаться.

Возможным подходом к распознаванию указанных вариантов может служить длительность интервала ожидания поступления дескрипторов из ППМ. Для этого можно использовать таймер, запрограммированный на интервал времени срабатывания, длительность которого связана с максимальным временем выполнения заданий.

Если после срабатывания таймера дескриптор данных не поступил, то произошел отказ ППМ, то есть необходимо повторное выполнение задания, которое формирует ожидаемый дескриптор данных.

Имя задания, которое не завершено, в данном случае находится в СВЗ. Для повторного выполнения задания достаточно переписать его имя из СВЗ в СГЗ. Заявка поступит на повторное выполнение с модифицированной маской, в соответствии с которой в ППМ будут формироваться только те данные, которые ранее не были получены.

Метод программно-аппаратной реконфигурации систем

Можно выделить три вида резервирования: поэлементное, общее и скользящее. Для систем с общей магистралью, построенных на однотипных процессорных модулях естественным является скользящее резервирование. Это объясняется не только универсальностью резерва (все модули одинаковы), но и возможностью подключения резерва в любую точку общей шины. Замена модуля не требует включения резервного модуля именно в то место шины, где находился отказавший модуль.

Для обеспечения однотипности модулей и, следовательно, простоты масштабирования целесообразно использовать аппаратные средства реконфигурации, которые распределены между процессорными модулями системы. В этом случае сохраняются основные преимущества модульных систем.

Выбор эффективного способа резервирования сопряжен с рядом трудностей, лежащих в области разработки процедур диагностирования, локализации и восстановления системы, которые бы минимально усложняли программы и вносили бы минимальное запаздывание в цикл управления.

Рассмотрим метод контроля и реконфигурации систем, который обладает следующими свойствами:

- вносит минимальную информационную избыточность (минимально усложняет программные средства);
- обеспечивает голосование и отключение неисправных модулей с использованием аппаратных средств для уменьшения расхода команд;
- автоматически производится реконфигурация системы (вместо отказавшего модуля вводится резервный);
- позволяет использовать любое число резервных модулей без усложнения схемы реконфигурации.

Метод базируется на контроле временных интервалов и событий. Выбор необходимых временных интервалов определяется режимом работы системы. Управление процессами (объектами) в реальном времени может быть реализовано в виде повторяющихся циклов. В каждом цикле управления вводится информация о состоянии процесса, затем осуществляется преобразование информации с целью определения необходимого воздействия на управляемый процесс. Полученные результаты используются для изменения параметров управления. На основании длительности цикла определяются необходимые временные интервалы для таймеров УПМ и ППМ.

Между процессорными модулями распределены аппаратные средства, предназначенные для ускорения процесса реконфигурации системы при отказе процессоров. Они представляют эстафетную цепочку передачи сигнала назначения нового УПМ при отказе текущего. Соответствующие логические цепи между соседними модулями показаны на рис. 1. В табл. 1 поясняются назначения некоторых разрядов КР, в том числе, которые используются при реконфигурации системы.

При инициализации системы первый ПМ назначается в качестве УПМ. Далее по цепочке могут располагаться РПМ, предварительно отключенные от общей шины. За ними располагаются ППМ, готовые к работе.

Таблица 1. Назначение разрядов командного регистра

Наименование разряда	Обозначение	Доступ со стороны локальной шины	Доступ со стороны системной шины
Готовность ППМ	ГП	Чтение / Запись	Чтение
Запуск программы ППМ	ЗП	Чтение / Запись	Чтение / Запись
Отключение ППМ	ОП	–	Запись
Отключение УПМ	ОУ	–	–
Неисправность ППМ	НП	–	–
Неисправность УПМ	НУ	Запись	–
Признак УПМ	ПУ	Чтение	–
Подключение РПМ	ПР	–	Запись

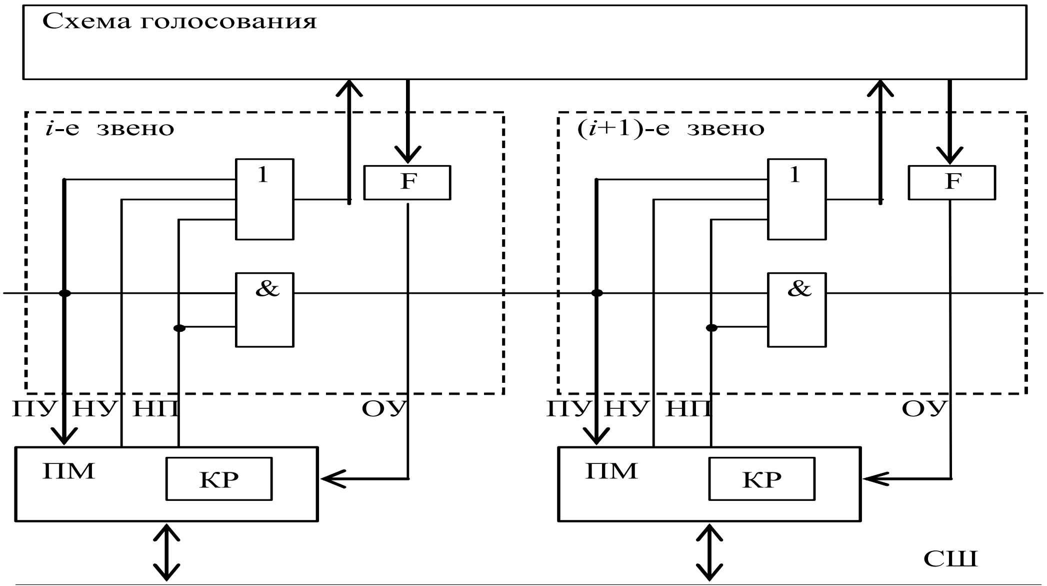


Рис. 1. Организация распределенных средств голосования процессоров

Работоспособность УПМ проверяется совместным голосованием работоспособных ППМ. Стратегия голосования выбирается, исходя из необходимого минимального числа работоспособных процессоров, которые, с учетом деградации системы, могут обеспечить решение поставленной задачи. Если система может выполнять свои функции при наличии g из n ППМ ($g<n$), то УПМ считается неисправным, если за его отключение проголосуют g или более процессорных модулей. В соответствии со значением g настраивается схема голосования, функции которой может выполнять пороговый элемент.

Один из простых алгоритмов работы системы состоит в следующем. В начале каждого цикла УПМ запускает все ППМ (сигнал ЗП в табл. 1) в широковещательном режиме, проверяя предварительно их готовность (сигнал ГП). Затем выполняет свою программу. ППМ запускают свой таймер и начинают выполнять свою программу, после выполнения которой устанавливают разряд ГП в КР и ожидают срабатывания своего таймера. Если при срабатывании таймера отсутствует сигнал ЗП, то ППМ голосует за отключение УПМ. Отключенные ПМ в голосовании участия не принимают. При наличии не менее g голосов эстафетная цепочка передает функции УПМ следующему по цепочке ПМ, который активизируется для работы в качестве УПМ. Каждое i-е звено эстафетной цепочки формирует выходной сигнал $ПУ_{i+1}$ в соответствии с переключательной функцией $ПУ_{i+1} = ПУ_i$ & $НП_i$. Если i-й процессор неисправен, то сигнал передается далее по цепочке и, кроме того, поступает на вход разряда ПУ регистра команд (i+1)-го процессора.

Работоспособность ППМ контролируется со стороны УПМ. Простой алгоритм контроля также связан со срабатыванием таймера в УПМ. Таймер запускается на время гарантированного выполнения программ в ППМ. По срабатыванию таймера УПМ проверяет в КР ППМ разряд ГП. При отсутствии признака ГП соответствующий ППМ отключается от системной шины, а вместо него подключается и запускается РПМ.

Процессоры затрачивают на взаимный контроль и голосование всего несколько команд на протяжении цикла управления. Если в цикле управления у процессоров

имеется оперативный запас времени, то они могут для контроля выполнять тестовую программу. Это позволяет повысить достоверность вычислений.

Рассмотренный программно-аппаратный метод реконфигурации позволяет по сравнению с программными методами, требующими при голосовании многократного обращения процессоров к общей памяти, ускорить процесс реконфигурации систем.

Заключение

Предложенный метод реализации вычислений в мультипроцессорных системах позволяет устранить ряд проблем, связанных с традиционным планированием вычислений. Существенно упрощается процесс подготовки задачи, поскольку нет необходимости на основе статического анализа выявлять параллельные процессы и заниматься их синхронизацией.

Подготовка задач не зависит от числа процессорных модулей в системе. Благодаря этому реконфигурация системы не приводит к необходимости повторной подготовки задач, что создает потенциальную возможность продолжать вычисления при отказе процессоров.

Система может работать в многопрограммном режиме, для чего не требуется обязательная регистрация всех задач перед началом счета. Начинать решение новой задачи можно в любой момент времени, независимо от состояния других задач. При этом благодаря управлению вычислениями со стороны потоков данных возможна реализация скрытого параллелизма задач, который трудно выявить на этапе статического анализа.

Исправление ошибок, связанных с отказом процессоров, осуществляется автоматически. Для этого достаточно на этапе разработки библиотеки функций предусмотреть возврат результатов согласно значению разрядов маски.

За счет применения аппаратных средств голосования и реконфигурации уменьшается расход команд при восстановлении системы. Все это позволяет уменьшить непроизводительные затраты времени, что создает предпосылки для ускорения вычислений, то есть повышения эффективности применения мультипроцессорных систем.

Литература

1. Silva J. Design of processing subsystems for Manchester data flow computer / J.Silva, J.Wood // IEEE Proc. N.Y. – 1981. – Vol. 128, N 5. – P. 218 – 224.
2. Watson R. A practical data flow computer / R.Watson, J.Guard // Computer. – 1982. – Vol. 15, N 2. – P. 51-57.
3. Коваленко А.Е. Отказоустойчивые микропроцессорные системы / А.Е.Коваленко, В.В.Гула. – К.: Техніка, 1986. – 150 с.
4. Жабин В.И. Метод распараллеливания процессов в вычислительных системах /В.И.Жабин // Вісник Національного технічного університету України “Київський політехнічний інститут”. Інформатика, управління та обчислювальна техніка. – 2000. – № 34. – С. 136-142.
5. Жабин В.И. Реализация параллельных процессов в вычислительных системах / В.И.Жабин // Искусственный интеллект. – 2002. – №3. – С. 235-241.
6. Жабин В.И. Реализация вычислений под управлением дескрипторов данных в мультипроцессорных системах / В.И.Жабин // Электронное моделирование. – 2003. – Т. 25, № 1. – С. 35-47.
7. Жабин В.И. Архитектура вычислительных систем реального времени / В.И.Жабин. – К.: ТОО “ВЕК+”, 2003. – 176 с.

Literatura

1. Silva J., Wood J.V. Design of processing subsystems for Manchester data flow computer // IEEE Proc. N.Y. – 1981. – Vol. 128, N 5. – P. 218 – 224.
2. Watson R., Guard J. A practical data flow computer // Computer. – 1982. – Vol. 15, N 2. – P. 51-57.

3. Kovalenko A.E., Gula V.V. Fail-safe microprocessor systems. – Kyiv: Technika, 1986. – 150 p. (in Russian)
4. Zhabin V.I. Method of process parallelizing in computer systems // NTUU "KPI" Reporter. Computer science, control and computer engineering. – 2000. – N 34. – P. 136-142. (in Russian)
5. Zhabin V.I. Parallel process realization in computer systems // Artificial intelligence. – 2002. – N 3. – P. 235-241. (in Russian)
6. Zhabin V.I. Realization of calculations controlled by the data descriptor flow in multiprocessor systems // Electronic simulation. – 2003. – V. 25, N 1. – P. 35-47. (in Russian)
7. Zhabin, V.I. The architecture of the real-time computer systems. – Kyiv: VEK+, 2003. – 176 p. (in Russian).

RESUME

V.I.Zhabin

Increase of parallel data processing effectiveness and control of real-time multiprocessor computer system failure safety

There is considered a problem of increase of coarse-grained parallelism realization effectiveness in multiprocessor systems built on homogeneous processors. Possibility to shorten problem resolution time realizing calculations controlled by data descriptor flow is shown. This saving of time is achieved by means of dynamic task distribution between processors, decrease of quantity of transmitted information, as well as implicit parallelism realization. The process of task preparation is simplified as it is not necessary to take into account task performance time, to look for parallel branches on the base of static analysis, to synchronize processes.

In the model of calculation control by means of data descriptor flow there is no need to take into account the quantity of processors during data preparation for processing in the system. The data are prepared only on the base of the task flow graph. This model enables to keep on calculations after processor failure if there are still some working processors in the system. A method of automatic correction of errors appeared because of task processing interruption is proposed. During next descriptor generation for interrupted tasks the data which were received before unit failure and do not need to be generated again is defined by means of masks.

There is proposed a hardware and software method of system reconfiguration which permits to shorten the list of instructions necessary for system recovery in comparison with the software method. For this purpose there is used hardware circuitry distributed between processors for voting and reconfiguration. All these means decrease non-productive time consumption and therefore shorten task resolution time.

В.И.Жабин

Повышение эффективности параллельной обработки данных и обеспечение отказоустойчивости мультипроцессорных вычислительных систем реального времени

Рассматривается задача повышения эффективности реализации крупнозернистого параллелизма в мультипроцессорных системах, построенных на однотипных процессорных модулях Показана возможность сокращения времени решения задач за счет использавания модели вычислений под управлением дескрипторов данных. Это достигается путем динамического распределения заданий между процессорными модулями, сокращения пересылаемых объемов информации, а также реализации скрытого параллелизма задач. Упрощается процесс подготовки задачи, поскольку нет необходимости учитывать длительность выполнения заданий, выявлять параллельные ветви на основе статического анализа, заниматься вопросом синхронизации процессов.

Модель управления вычислениями со стороны потока дескрипторов данных не требует при подготовке данных для обработки учета числа процессоров в системе. Данные подготавливаются только на основе потокового графа задачи. Это создает потенциальную возможность продолжать вычисления при отказе процессоров, если работоспособные процессоры в системе остаются. Предложен метод автоматического исправления ошибок, которые могут возникать из-за возможного прерывания выполнения заданий. При повторном формировании дескрипторов для прерваных заданий с помощью масок указывается, какие данные были получены до отказа процессорного модуля и не требуют повторного формирования.

Предложен программно-аппаратный метод реконфигурации систем, позволяющий сократить расход команд по сравнению с программным методом при восстановлении работоспособности систем. Для этого используются распределенные между процессорами аппаратные схемы голосования и реконфигурации. Все это уменьшает непроизводительные затраты времени, что создает предпосылки для ускорения решения задач.

Поступила в редакцию 03.07.2015